

-\$25  
Symbol  
10C1  
10-C  
10-D  
10-F  
10-S  
KICL  
  
KILL  
KILL  
LB-E  
LB-D  
LB-F  
LB-H  
LB-L  
LOCK  
LOCK  
LOCK  
LOC  
LOC  
L-CC  
L-CC  
L-DA  
L-DA  
MAIN  
MAKE  
MAKE  
MAKE  
MAKE  
MAKE  
MAKE  
MAKE  
MAP  
MAP  
MAP  
MAP  
MAP  
MAP

\*\*FILE\*\*ID\*\*DELETE

C 8

DDDDDDDD DDDDDDDDD EEEEEEEEEE EEEEEE EEEEEE EEEEEE  
DD DD EE DD EE LL  
DD DD EE DD EE LL  
DD DD EEEEEEEE DD EE LL  
DD DD EEEEEEEE DD EE LL  
DD DD EE DD EE LL  
DD DD EE DD EE LL  
DD DD EE DD EE LL  
DD DD EE DD EE LL  
DDDDDDDD DDDDDDDDD EEEEEEEEEE LLLLLLLLLL EEEEEEEEEE TT TT EEEEEEEEEE  
DDDDDDDD DDDDDDDDD EEEEEEEEEE LLLLLLLLLL EEEEEEEEEE TT TT EEEEEEEEEE

LL II IIII SSSSSSSS  
LL II IIII SSSSSSSS  
LL SS SS SS  
LLLLLLLLL LLLL LLLL SSSSSSSS SSSSSSSS

DEI  
VO

```
1 0001 0 MODULE DELETE (
2 0002 0           LANGUAGE (BLISS32),
3 0003 0           IDENT = 'V04-000'
4 0004 0           ) =
5 0005 1 BEGIN
6
7 0007 1 ****
8 0008 1 *
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 * ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 * TRANSFERRED.
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 * CORPORATION.
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 ****
30 0030 1 *
31 0031 1 ++
32 0032 1 *
33 0033 1 FACILITY: F11ACP Structure Level 2
34 0034 1 *
35 0035 1 ABSTRACT:
36 0036 1 *
37 0037 1     This routine performs the DELETE function.
38 0038 1 *
39 0039 1 ENVIRONMENT:
40 0040 1 *
41 0041 1     STARLET operating system, including privileged system services
42 0042 1     and internal exec routines.
43 0043 1 *
44 0044 1 --
45 0045 1 *
46 0046 1 *
47 0047 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 1-Apr-1977
48 0048 1 *
49 0049 1 MODIFIED BY:
50 0050 1 *
51 0051 1     V03-024 CDS0015 Christian D. Saether 14-Aug-1984
52 0052 1     Modify handling of extension fcbs.
53 0053 1 *
54 0054 1     V03-023 CDS0014 Christian D. Saether 10-Aug-1984
55 0055 1     Clear directory flag in header prior to actually
56 0056 1     deleting file so that extra checks against deleting
57 0057 1     a directory can be made in delete_file.
```

58 0058 1  
59 0059 1  
60 C060 1  
61 0061 1  
62 0062 1  
63 0063 1  
64 0064 1  
65 0065 1  
66 0066 1  
67 0067 1  
68 0068 1  
69 0069 1  
70 0070 1  
71 0071 1  
72 0072 1  
73 0073 1  
74 0074 1  
75 0075 1  
76 0076 1  
77 0077 1  
78 0078 1  
79 0079 1  
80 0080 1  
81 0081 1  
82 0082 1  
83 0083 1  
84 0084 1  
85 0085 1  
86 0086 1  
87 0087 1  
88 0088 1  
89 0089 1  
90 0090 1  
91 0091 1  
92 0092 1  
93 0093 1  
94 0094 1  
95 0095 1  
96 0096 1  
97 0097 1  
98 0098 1  
99 0099 1  
100 0100 1  
101 0101 1  
102 0102 1  
103 0103 1  
104 0104 1  
105 0105 1  
106 0106 1  
107 0107 1  
108 0108 1  
109 0109 1  
110 0110 1  
111 0111 1  
112 0112 1  
113 0113 1  
114 0114 1

V03-022 CDS0013 Christian D. Saether 7-Aug-1984  
Wipe out directory index if there is one when  
deleting the fcb. Use common routine to delete fcb.

V03-021 CDS0012 Christian D. Saether 6-Aug-1984  
Sense of test in CDS0011 to fix access arbitration  
on exclusively accessed file was wrong. Fix it.

V03-020 CDS0011 Christian D. Saether 31-July-1984  
Remove local declaration of get\_map\_pointer linkage.  
Fix access arbitration check to allow deletion if  
we have it accessed exclusively readonly.

V03-019 LMP0275 L. Mark Pilant, 23-Jul-1984 14:19  
Don't try to delete an uninitialized ACL.

V03-018 ACG0427 Andrew C. Goldstein, 8-May-1984 13:32  
Write audit record for file about to be deleted

V03-017 CDS0010 Christian D. Saether 4-May-1984  
Remember to release access lock in MARKDEL\_FCB if  
we get rid of the fcb there.

V03-016 CDS0009 Christian D. Saether 19-Apr-1984  
Changes to restore compatible (with V3) delete behavior.

V03-015 ACG0415 Andrew C. Goldstein, 5-Apr-1984 21:31  
Interface change to ACL\_DELETEACL

V03-014 ACG0412 Andrew C. Goldstein, 22-Mar-1984 18:21  
Implement agent access mode support; add access mode to  
check protection call

V03-013 ACG0408 Andrew C. Goldstein, 20-Mar-1984 17:35  
Make APPLY\_RVN and DEFAULT\_RVN macros; remove delete logger

V03-012 CDS0008 Christian D. Saether 23-Feb-1984  
Change references to FLUSH\_LOCK\_BASIS to WRITE\_DIRTY.  
Checksum header and mark dirty when only marking  
for delete and not actually deleting file.  
Modify call to ACL\_DELETEACL.

V03-011 CDS0007 Christian D. Saether 17-Jan-1984  
Modify interface to APPLY\_RVN.

V03-010 CDS0006 Christian D. Saether 27-Dec-1983  
Use BIND\_COMMON macro.

V03-009 CDS0005 Christian D. Saether 13-Dec-1983  
Move all OWN data declarations to the  
COMMON module.

V03-008 LMP0178 L. Mark Pilant, 8-Dec-1983 14:22  
Fix a bug that caused paged pool to be lost when deleting  
an unaccessed file.

: 115 0115 1 | V03-007 ACG0368 Andrew C. Goldstein, 4-Nov-1983 14:24  
: 116 0116 1 | Handle short ident areas in back link file name check  
: 117 0117 1 |  
: 118 0118 1 | V03-006 CDS0004 Christian D. Saether 14-Sep-1983  
: 119 0119 1 | Modify SERIAL FILE interface.  
: 120 0120 1 | Call RELEASE\_SERIAL\_LOCK to dequeue.  
: 121 0121 1 |  
: 122 0122 1 | V03-005 CDS0003 Christian D. Saether 6-May-1983  
: 123 0123 1 | Call SERIAL\_FILE to interlock file processing.  
: 124 0124 1 | Remove SWITCH\_VOLUME and SEARCH\_FCB calls in DELETE  
: 125 0125 1 | routine because they are called from MARK\_DELETE now.  
: 126 0126 1 | Call FLUSH\_FID at the end of MARK\_DELETE so that  
: 127 0127 1 | file processing interlock can be released. This is  
: 128 0128 1 | necessary because of the call from CREATE using  
: 129 0129 1 | secondary context.  
: 130 0130 1 |  
: 131 0131 1 | V03-004 ACG0323 Andrew C. Goldstein, 12-Apr-1983 16:12  
: 132 0132 1 | Fix passing of result string buffer  
: 133 0133 1 |  
: 134 0134 1 | V03-003 CDS0002 Christian D. Saether 7-Apr-1983  
: 135 0135 1 | Modifications to correctly arbitrate delete actions  
: 136 0136 1 | in a cluster.  
: 137 0137 1 |  
: 138 0138 1 | V03-002 ACG0323 Andrew C. Goldstein, 25-Mar-1983 16:29  
: 139 0139 1 | Erase back link when matching directory entry is removed  
: 140 0140 1 |  
: 141 0141 1 | V03-001 LMP0059 L. Mark Pilant, 27-Dec-1982 8:14  
: 142 0142 1 | Always create an FCB for a file header. This eliminates a  
: 143 0143 1 | lot of special case FCB handling.  
: 144 0144 1 |  
: 145 0145 1 | V02-006 ACG0249 Andrew C. Goldstein, 29-Dec-1981 13:58  
: 146 0146 1 | Use DATA block type to read directory block  
: 147 0147 1 |  
: 148 0148 1 | V02-005 ACG0227 Andrew C. Goldstein, 24-Nov-1981 22:45  
: 149 0149 1 | Protect directory files from deletion  
: 150 0150 1 |  
: 151 0151 1 | V02-004 ACG0167 Andrew C. Goldstein, 16-Apr-1980 19:25  
: 152 0152 1 | Previous revision history moved to F11B.REV  
: 153 0153 1 | \*\*  
: 154 0154 1 |  
: 155 0155 1 |  
: 156 0156 1 LIBRARY 'SYSSLIBRARY:LIB.L32';  
: 157 0157 1 REQUIRE 'SRCS:FCPDEF.B32';  
: 158 1148 1 |  
: 159 1149 1 |  
: 160 1150 1 FORWARD ROUTINE  
: 161 1151 1 DELETE : L\_NORM, ! main delete function  
: 162 1152 1 MARK\_DELETE : L\_NORM NOVALUE ! mark file for delete  
: 163 1153 1 MARKDEL\_FCB : L\_NORM, ! mark FCB of file for delete  
: 164 1154 1 DELETE\_HANDLER : L\_NORM; ! condition handler for delete function

```
: 166      1155 1 GLOBAL ROUTINE DELETE : L_NORM =
: 167      1156 1
: 168      1157 1 ++
: 169      1158 1
: 170      1159 1 FUNCTIONAL DESCRIPTION:
: 171      1160 1
: 172      1161 1 This routine performs the remove and mark for delete functions.
: 173      1162 1
: 174      1163 1 CALLING SEQUENCE:
: 175      1164 1     DELETE ()
: 176      1165 1
: 177      1166 1 INPUT PARAMETERS:
: 178      1167 1     NONE
: 179      1168 1
: 180      1169 1 IMPLICIT INPUTS:
: 181      1170 1     IO_PACKET: I/O packet in process
: 182      1171 1
: 183      1172 1 OUTPUT PARAMETERS:
: 184      1173 1     PRIMARY_FCB: FCB of file
: 185      1174 1
: 186      1175 1 IMPLICIT OUTPUTS:
: 187      1176 1     NONE
: 188      1177 1
: 189      1178 1 ROUTINE VALUE:
: 190      1179 1     1
: 191      1180 1
: 192      1181 1 SIDE EFFECTS:
: 193      1182 1     directory entry removed
: 194      1183 1     file marked for delete or deleted
: 195      1184 1
: 196      1185 1 --+
: 197      1186 1
: 198      1187 2 BEGIN
: 199      1188 2
: 200      1189 2 LOCAL
: 201      1190 2     ABD      : REF BBLOCKVECTOR [,ABD$C_LENGTH],
: 202      1191 2                   buffer descriptors
: 203      1192 2     FIB      : REF BBLOCK,          FIB
: 204      1193 2     RESULT_LENGTH,    length of name string from directory
: 205      1194 2     RESULT       : VECTOR [FILENAME_LENGTH+6, BYTE];
: 206      1195 2                   ! File name string from directory
: 207      1196 2
: 208      1197 2     BIND_COMMON;
: 209      1198 2
: 210      1199 2     EXTERNAL ROUTINE
: 211      1200 2     GET_FIB      : L_NORM,        ! get FIB of request
: 212      1201 2     FIND         : L_NORM;        ! find name in directory
: 213      1202 2
: 214      1203 2
: 215      1204 2     ! First find the buffer descriptor, FIB, FCB, etc. then remove the
: 216      1205 2     ! directory entry.
: 217      1206 2     !
: 218      1207 2
: 219      1208 2     ! pointer to buffer descriptors
: 220      1209 2     ABD = .BBBLOCK [.IO_PACKET[IRPSL_SVAPTE], AIB$L_DESCRIPTOR];
: 221      1210 2     FIB = GET_FIB (.ABD);
: 222      1211 2
```

```

223      1212 2 IF .CURRENT_VCB[VCBSV_NOALLOC]
224      1213 2 THEN ERR_EXIT ($SS_WRTLOCK);
225      1214 ! If a directory ID is present, do a directory search first and remove
226      1215 ! the directory entry.
227      1216 !
228      1217 !
229      1218 !
230      1219 RESULT_LENGTH = 0;
231      1220 IF .CLEANUP_FLAGS[CLF_DIRECTORY]
232      1221 THEN FIND (.ABD, .FIB, 1, RESULT_LENGTH, RESULT);
233      1222 !
234      1223 ! If there is a file open on the channel, check the file ID returned by the
235      1224 ! FIND against that of the open file. If they do not match, treat the file
236      1225 ! as if it were not open.
237      1226 !
238      1227 !
239      1228 IF .PRIMARY_FCB NEQ 0
240      1229 THEN
241      1230 BEGIN
242      1231 IF .PRIMARY_FCB[FCBSW_FID_NUM] NEQ .FIB[FIBSW_FID_NUM]
243      1232 OR .PRIMARY_FCB[FCBSW_FID_SEQ] NEQ .FIB[FIBSW_FID_SEQ]
244      1233 THEN CURRENT_WINDOW = 0;
245      1234 END;
246      1235 !
247      1236 ! Now actually mark the file for delete if requested.
248      1237 !
249      1238 !
250      1239 MARK_DELETE (.FIB, .BBLOCK [IO_PACKET[IRPSW_FUNC], IOSV_DELETE], .RESULT_LENGTH, RESULT);
251      1240 !
252      1241 RETURN 1;
253      1242 !
254      1243 1 END:                                ! end of routine DELETE

```

.TITLE DELETE  
.IDENT \V04-000\

.EXTRN GET\_FIB, FIND

.PSECT SCODE\$,NOWRT,2

				000C 00000		.ENTRY	DELETE, Save R2,R3	: 1155
		5E	A4	AE 9E 00002		MOVAB	-92(SP), SP	: 1209
		50	90	AA D0 00006		MOVL	-112(BASE), R0	: 1210
		53	2C	B0 D0 0000A		MOVL	344(R0), ABD	: 1211
				53 DD 0000E		PUSHL	ABD	: 1212
		0000G	CF	01 FB 00010		CALLS	#1, GET_FIB	: 1213
				52 50 D0 00015		MOVL	R0, FIB	
		05	0B	98 AA D0 00018		MOVL	-104(BASE), R0	
			A0	04 E1 0001C		BBC	#4 11(R0), 1\$	
				025C 8F BF 00021		CHMU	#604	
				04 04 00025		RET		
		11	6A	6E D4 00026	1\$:	CLRL	RESULT_LENGTH	: 1219
				06 E1 00028		BBC	#6, (BASE), 2\$	: 1220
				04 AE 9F 0002C		PUSHAB	RESULT	: 1221
				04 AE 9F 0002F		PUSHAB	RESULT_LENGTH	
				01 DD 00032		PUSHL	#1	

			52 DD 00034	PUSHL	FIB	
			53 DD 00036	PUSHL	ABD	
		0000G CF 50	08 AA DD 0003D	CALLS	#5, FIND	
			11 13 00041	MOVL	8(BASE), R0	1228
		04 A2 24	A0 B1 00043	BEQL	4\$	
			07 12 00048	CMPW	36(R0), 4(FIB)	1231
		06 A2 26	A0 B1 0004A	BNEQ	38	
			03 13 0004F	CMPW	38(R0), 6(FIB)	1232
			0C AA D4 00051	BEQL	4\$	
			04 AE 9F 00054	CLRL	12(BASE)	1233
			4\$:	PUSHAB	RESULT	1239
			04 AE DD 00057	PUSHL	RESULT LENGTH	
			90 AA DD 0005A	MOVL	-112(BASE), R0	
		7E 21 A0	01 00 EF 0005E	EXTZV	#0, #1, 33(R0), -(SP)	
			52 DD 00064	PUSHL	FIB	
		0000V CF 50	04 FB 00066	CALLS	#4, MARK_DELETE	
			01 DD 0006B	MOVL	#1, R0	1241
			04 0006E	RET		1243

; Routine Size: 111 bytes. Routine Base: \$CODE\$ + 0000

: 256 1244 1 GLOBAL ROUTINE MARK\_DELETE (FIB, DO\_DELETE, RESULT\_LENGTH, RESULT) : L\_NORM NOVALUE =  
: 257 1245 1  
: 258 1246 1 ++  
: 259 1247 1  
: 260 1248 1 FUNCTIONAL DESCRIPTION:  
: 261 1249 1  
: 262 1250 1 This routine marks the indicated file for delete and deletes it  
: 263 1251 1 if it is not accessed.  
: 264 1252 1  
: 265 1253 1 CALLING SEQUENCE:  
: 266 1254 1 MARK\_DELETE (ARG1, ARG2, ARG3, ARG4)  
: 267 1255 1  
: 268 1256 1 INPUT PARAMETERS:  
: 269 1257 1 ARG1: address of FIB  
: 270 1258 1 ARG2: 1 to actually delete the file  
: 271 1259 1 0 to only remove the directory entry  
: 272 1260 1 ARG3: length of name string from directory operation  
: 273 1261 1 ARG4: address of name string  
: 274 1262 1  
: 275 1263 1 IMPLICIT INPUTS:  
: 276 1264 1 NONE  
: 277 1265 1  
: 278 1266 1 OUTPUT PARAMETERS:  
: 279 1267 1 NONE  
: 280 1268 1  
: 281 1269 1 IMPLICIT OUTPUTS:  
: 282 1270 1 NONE  
: 283 1271 1  
: 284 1272 1 ROUTINE VALUE:  
: 285 1273 1 NONE  
: 286 1274 1  
: 287 1275 1 SIDE EFFECTS:  
: 288 1276 1 file marked for delete or deleted  
: 289 1277 1  
: 290 1278 1 --  
: 291 1279 1  
: 292 1280 2 BEGIN  
: 293 1281 2  
: 294 1282 2 BUILTIN  
: 295 1283 2 FP;  
: 296 1284 2  
: 297 1285 2 MAP  
: 298 1286 2 FIB : REF BBLOCK; ! FIB  
: 299 1287 2  
: 300 1288 2 GLOBAL REGISTER  
: 301 1289 2 COUNT = 6. : map pointer count  
: 302 1290 2 LBN = 7. : map pointer LBN  
: 303 1291 2 MAP\_POINTER = 8; : pointer to file header map area  
: 304 1292 2  
: 305 1293 2 LOCAL  
: 306 1294 2 Curr\_LkMode, : mode access lock currently held at.  
: 307 1295 2 EOF : end of file VBN of file  
: 308 1296 2 BUFFER : REF VECTOR [.WORD], ! buffer address of block read  
: 309 1297 2 FCB : REF BBLOCK, ! FCB of file  
: 310 1298 2 HEADER : REF BBLOCK, ! file header  
: 311 1299 2 IDENT\_AREA : REF BBLOCK, ! header's ident area  
: 312 1300 2 TEMP\_FID : BBLOCK [FIDSC\_LENGTH], ! temp copy of file ID

: 313 1301 2 FCB\_CREATED,  
314 1302 2 NEW\_HEADER : REF BBLOCK, : Flag indicating new FCB created  
315 1303 2 ARGCLST : REF BBLOCK; : Address of extension header  
316 1304 2 : pointer to audit block entries  
317 1305 2 BIND\_COMMON;  
318 1306 2  
319 13C7 2 EXTERNAL ROUTINE  
320 1308 2 REBLD\_PRIM\_FCB : L\_NORM NOVALUE, : rebuild primary fcb from header  
321 1309 2 BUILD\_EXT\_FCB : L\_NORM NOVALUE, : build extension fcb chain  
322 1310 2 KILL\_DINDEX : L\_NORM NOVALUE, : delete directory index  
323 1311 2 KILL\_BUFFERS : L\_NORM NOVALUE, : kill directory buffers  
324 1312 2 NUKE\_HEAD\_FCB : L\_NORM NOVALUE, : cleanup and delete prim fcb  
325 1313 2 DEL\_EXTFCB : L\_NORM, : delete extension FCBS.  
326 1314 2 ARBITRATE\_ACCESS : [ JSB\_2ARGS, : determine allowed file access  
327 1315 2 CONV\_ACLOCK : L\_NORM, : convert file access lock.  
328 1316 2 WRITE\_DIRTY : L\_NORM, : write back modified buffers.  
329 1317 2 SERIAL\_FILE : L\_NORM, : interlock file processing  
330 1318 2 RELEASE\_SERIAL\_LOCK : L\_NORM NOVALUE,  
331 1319 2 SWITCH\_VOLUME : L\_NORM, : switch context to desired volume  
332 1320 2 SEARCH\_FCB : L\_NORM, : search FCB list  
333 1321 2 CREATE\_FCB : L\_NORM, : create an FCB  
334 1322 2 READ\_HEADER : L\_NORM, : read file header  
335 1323 2 CHECK\_PROTECT : L\_NORM, : check file protection  
336 1324 2 WRITE\_AUDIT : L\_NORM, : write audit record  
337 1325 2 GET\_MAP\_POINTER : L\_MAP\_POINTER, : get file header map pointer  
338 1326 2 READ\_BLOCK : L\_NORM, : read a disk block  
339 1327 2 INVALIDATE : L\_NORM, : invalidate block buffer  
340 1328 2 MARK\_DIRTY : L\_NORM, : mark buffer for write-back  
341 1329 2 DELETE\_FILE : L\_NORM, : delete the file  
342 1330 2 CHECKSUM : L\_NORM; : checksum file header  
343 1331 2  
344 1332 2  
345 1333 2 : Find the FCB, if any, and then read the header. Reading the header is done  
346 1334 2 under a condition handler that quietly exits with success if errors are  
347 1335 2 encountered. Thus, deleting a bad file header succeeds quietly.  
348 1336 2  
349 1337 2  
350 1338 2 SWITCH\_VOLUME (.FIB[FIB\$W\_FID\_RVN]);  
351 1339 2  
352 1340 2 : Serialize further processing on this file.  
353 1341 2  
354 1342 2  
355 1343 2 PRIM\_LCKINDX = SERIAL\_FILE (FIB [FIB\$W\_FID]);  
356 1344 2  
357 1345 2 FCB = SEARCH\_FCB (FIB[FIB\$W\_FID]);  
358 1346 2 SAVE\_STATUS = .USER\_STATUS;  
359 1347 2 .FP = DELETE\_HANDLER;  
360 1348 2 HEADER = READ\_HEADER (FIB[FIB\$W\_FID], .FCB);  
361 1349 2 .FP = 0;  
362 1350 2  
363 1351 2 : If this is a real delete, proceed with it.  
364 1352 2  
365 1353 2  
366 1354 2 IF .DO\_DELETE  
367 1355 2 THEN BEGIN  
368 1356 2  
369 1357 2

```
; 370      1358 3 | Check that the file is not a reserved file (FID less than
; 371      1359 3 | .CURRENT_VCB[VCB$B_RESERVED]).  
;  
; 372      1360 3 |  
; 373      1361 3 |  
; 374      1362 3 | IF .FIB[FIB$W_FID_NUM] LEOU .CURRENT_VCB[VCB$B_RESERVED]  
; 375      1363 3 | AND .FIB[FIB$B_FID_NMX] EQL 0  
; 376      1364 3 | THEN ERR_EXIT TSS$ NOPRIV);  
;  
; 377      1365 3 |  
; 378      1366 3 | At this point, build the necessary FCB chain to allow the ACL to be built.  
;  
; 379      1367 3 |  
; 380      1368 3 | FCB_CREATED = 0;  
; 381      1369 3 | IF .FCB EQL 0  
; 382      1370 3 | THEN  
; 383      1371 4 | BEGIN  
; 384      1372 4 |   FCB_CREATED = 1;  
; 385      1373 4 |   FCB = KERNEL_CALL (CREATE_FCB, .HEADER);  
; 386      1374 3 | END;  
; 387      1375 3 | PRIMARY_FCB = .FCB;           ! Record FCB for external use  
;  
; 388      1376 3 |  
; 389      1377 3 | If the file is multi-header, read in the extension headers and create  
; 390      1378 3 | extension FCB's. Finally, read back the primary header.  
;  
; 391      1379 3 |  
; 392      1380 3 |  
; 393      1381 3 | IF .FCB_CREATED  
; 394      1382 3 | THEN  
; 395      1383 3 |   BUILD_EXT_FCB (.HEADER)  
; 396      1384 3 | ELSE  
; 397      1385 3 |   IF .FCB [FCBSV_STALE]  
; 398      1386 3 |   THEN  
; 399      1387 4 |     BEGIN  
;  
; 400      1388 4 |       REBLD_PRIM_FCB (.FCB, .HEADER);  
;  
; 401      1389 4 |       BUILD_EXT_FCB (.HEADER);  
;  
; 402      1390 4 |     END;  
;  
; 403      1391 4 |  
; 404      1392 4 |  
; 405      1393 4 |  
; 406      1394 4 |  
;  
; 407      1395 3 | Check file protection. Check if the file is write accessed by someone  
; 408      1396 3 | else and not the deleter.  
;  
; 409      1397 3 |  
;  
; 410      1398 3 |  
;  
; 411      1399 3 | CHECK_PROTECT (DELETE_ACCESS, .HEADER, .FCB,  
; 412      1400 3 |   MAXU (.IO_PACKET[IRPSV_MODE], .FIB[FIB$B_AGENT_MODE]));  
;  
; 413      1401 3 |  
; 414      1402 3 | If the file is identified as a directory, check to see if it is empty.  
; 415      1403 3 | Non-empty directories cannot be deleted under any circumstances.  
; 416      1404 3 | The check for emptiness is done by (1) checking for a length of  
; 417      1405 3 | 1 block, and (2) reading the block and looking for the data pattern of  
; 418      1406 3 | an empty directory block.  
;  
; 419      1407 3 |  
;  
; 420      1408 3 |  
; 421      1409 3 | IF .HEADER[FH2$V_DIRECTORY]  
; 422      1410 3 | THEN  
; 423      1411 4 | BEGIN  
;  
; 424      1412 4 |   EOF = ROT (.BBLOCK [HEADER[FH2$W_RECATTR], FAT$L_EFBLOCK], 16);  
; 425      1413 4 |   IF .EOF NEQ 0  
; 426      1414 4 |     AND .BBLOCK [HEADER[FH2$W_RECATTR], FAT$W_FFBYTE] EQL 0
```

```
: 427      1415 4   THEN EOF = EOF - 1;
: 428      1416 4   IF .EOF LEQU 1
: 429      1417 4   THEN
: 430      1418 5   BEGIN
: 431      1419 5   MAP_POINTER = .HEADER + .HEADER[FH2SB_MPOFFSET] + 2;
: 432      1420 5   GET_MAP_POINTER ();
: 433      1421 5   BUFFER = READ_BLOCK (.LBN, 1, DATA_TYPE);
: 434      1422 5   IF .BUFFER[0] NEQ 65535
: 435      1423 5   THEN ERR_EXIT (SSS_DIRNOTEMPTY);
: 436      1424 5   INVALIDATE (.BUFFER);
: 437      1425 5   END
: 438      1426 4   ELSE ERR_EXIT (SSS_DIRNOTEMPTY);
: 439      1427 4
: 440      1428 3
: 441      1429 3
: 442      1430 3   : Check if a security audit record is to be written for this file.
: 443      1431 3   If so, now is the last time to do it. ('Morituri te salutamus!')
: 444
: 445
: 446      1434 3   ARGLIST = AUDIT_ARGLIST;
: 447      1435 3   DECR J FROM MAX_AUDIT_COUNT TO 1
: 448      1436 3   DO
: 449      1437 4   BEGIN
: 450      1438 4   IF .ARGLIST[AUDIT_TYPE] NEQ 0
: 451      1439 4   AND .BBLOCK [ARGLIST[AUDIT_FID], FIDSW_NUM] EQL .FCB[FCBSW_FID_NUM]
: 452      1440 4   AND .BBLOCK [ARGLIST[AUDIT_FID], FIDSW_SEQ] EQL .FCB[FCBSW_FID_SEQ]
: 453      1441 4   AND .BBLOCK [ARGLIST[AUDIT_FID], FIDSW_RVN] EQL .FCB[FCBSW_FID_RVN]
: 454      1442 4   THEN
: 455      1443 5   BEGIN
: 456      1444 5   WRITE_AUDIT (.ARGLIST);
: 457      1445 5   HEADER = .FILE_HEADER;
: 458      1446 5   EXITLOOP 0;
: 459      1447 4
: 460      1448 4   ARGLIST = .ARGLIST + AUDIT_LENGTH;
: 461      1449 3
: 462
: 463      1450 3   : Remember current lock mode to be restored later, if necessary.
: 464
: 465
: 466      1454 3   CURR_LKMODE = .FCB [FCBSB_ACCLKMODE];
: 467
: 468      1456 3   : Make access checks.
: 469      1457 3   If we have the file accessed, we may delete it as long as we have
: 470      1458 3   write access ourselves (whether there are other writers or not).
: 471      1459 3   In all other cases, no other writers are allowed.
: 472
: 473
: 474      1462 3   IF .CURRENT_WINDOW NEQ 0
: 475      1463 3   THEN
: 476      1464 4   BEGIN
: 477      1465 4   IF NOT .CURRENT_WINDOW [WCBSV_WRITE]
: 478      1466 4   AND NOT .FCB [FCBSV_EXCL]
: 479      1467 4   THEN
: 480      1468 4   IF NOT ARBITRATE_ACCESS (FIBSM_NOWRITE, .FCB)
: 481      1469 4   THEN
: 482      1470 5   END
: 483      1471 4   ERR_EXIT (SSS_ACCONFLICT)
```

```
484      1472 3    ELSE
485      1473 3      IF NOT ARBITRATE_ACCESS (FIBSM_NOWRITE, .FCB)
486      1474 3        THEN
487      1475 3          ERR_EXIT (SSS_ACCONFLICT);
488      1476 3
489      1477 3      [CLEANUP_FLAGS[CLF_REENTER] = 0;           ! from now on deletion proceeds
490      1478 3
491      1479 3
492      1480 3      ! Mark the file for delete. If the file is not accessed, then proceed to
493      1481 3        actually delete it.
494      1482 3      In addition, if this is a directory file, clear the directory flag in
495      1483 3        the header and clean out cached directory data blocks now.
496      1484 3      Clearing the directory flag in the header allows us to be defensive
497      1485 3        against accidental directory deletion in delete_file.
498      1486 3
499      1487 3
500      1488 3      HEADER[FH2SV_MARKDEL] = 1;
501      1489 3
502      1490 3      IF TESTBITS(HEADER [FH2$V_DIRECTORY])
503      1491 3        THEN
504      1492 3          KILL_BUFFERS (1, .FCB [FCBSL_LOCKBASIS]);
505      1493 3
506      1494 3      IF MARKDEL_FCB (.FCB)
507      1495 3        THEN
508      1496 3          DELETE_FILE (.FIB, .HEADER)
509      1497 3        ELSE
510      1498 4          BEGIN
511      1499 4            CHECKSUM (.HEADER);
512      1500 4            MARK_DIRTY (.HEADER);
513      1501 3          END;
514      1502 3
515      1503 3      ! The access lock conversion routine is called to:
516      1504 3        1) restore the previous lock mode
517      1505 3        2) dequeue the access lock entirely if the refcnt is zero.
518      1506 3        3) if the lock was granted exclusive, either restore or dequeue the
519      1507 3          lock and store the value block.
520      1508 3
521      1509 3
522      1510 3      CONV_ACLOCK (.CURR_LKMODE, .FCB);
523      1511 3
524      1512 3      IF .FCB [FCBSW_REFCNT] EQ 0
525      1513 3        THEN
526      1514 4          BEGIN
527      1515 4            IF .FCB [FCBSL_DIRINDX] NEQ 0
528      1516 4              THEN
529      1517 4                  KILL_DINDEX (.FCB);
530      1518 4
531      1519 4      DEL_EXTFCB (.FCB);
532      1520 4      NUKE_HEAD_FCB (.FCB);
533      1521 3
534      1522 3
535      1523 3      IF .PRIMARY_FCB EQ .FCB THEN PRIMARY_FCB = 0;
536      1524 3      IF .DIR_FCB EQ .FCB THEN DIR_FCB = 0;
537      1525 3
538      1526 3      END      ! of we really do want to delete the file.
539      1527 3
540      1528 3      ! Otherwise we are just removing a directory entry. If the file name
```

```

541 1529 3 | end back link in the header match the directory, erase the back
542 1530 3 | link.
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578

1531 3 | ! and back link in the header match the directory, erase the back
1532 3 | link.
1533 3 |
1534 3 | ELSE
1535 3 | BEGIN
1536 3 | CHSMOVE (FIDSC_LENGTH, HEADER[FH2SW_BACKLINK], PREV_LINK);
1537 3 | CHSMOVE (FIDSC_LENGTH, HEADER[FH2SW_BACKLINK], TEMP_FID);
1538 3 | APPLY_RVN (TEMP_FID[FDSW_RVN], .CURRENT_RVN);
1539 3 | IDENT_AREA = .HEADER + .HEADER[FH2SB_IDOFFSET]*2;
1540 3 | CHSCOPY (FI2SS_FILENAME, IDENT_AREA[FI2ST_FILENAME],
1541 3 | , FILENAME_LENGTH+6, PREV_INAME);
1542 3 | IF .HEADER[FH2SB_MPOFFSET] = .HEADER[FH2SB_IDOFFSET]
1543 3 | GEQU ($BYTEOFFSET(FI2ST_FILENOEXT) + FI2SS_FILENOEXT) / 2
1544 3 | THEN
1545 3 | CHSMOVE (FI2SS_FILENOEXT, IDENT_AREA[FI2ST_FILENOEXT],
1546 3 | , PREV_INAME[FI2SS_FILENAME]);
1547 3 | IF CHSEQL (FIDSC_LENGTH, FIB[FIBSW_DID], FIDSC_LENGTH, TEMP_FID)
1548 3 | AND CHSEQL (.RESULT_LENGTH, .RESULT,
1549 3 | FI2SS_FILENAME+FI2SS_FILENOEXT, PREV_INAME, ' ')
1550 3 | THEN
1551 3 | BEGIN
1552 3 | HEADER[FH2SW_BK_FIDNUM] = 0;
1553 3 | HEADER[FH2SW_BK_FIDSEQ] = 0;
1554 3 | HEADER[FH2SW_BK_FIDRVN] = 0;
1555 3 | CLEANUP_FLAGS[CC_FIXLINK] = 1;
1556 3 | CHECKSUM (.HEADER);
1557 3 | MARK_DIRTY (.HEADER);
1558 3 | END;
1559 2 | END;
1560 2 | WRITE_DIRTY (.LB_BASIS [.PRIM_LCKINDEX]);
1561 2 | RELEASE_SERIAL_LOCK (.PRIM_LCKINDEX);
1562 2 | PRIM_LCKINDEX = 0;
1563 2 |
1564 2 |
1565 2 |
1566 1 | END;

```

! end of routine MARK\_DELETE

.EXTRN	REBLD_PRIM_FCB, BUILD_EXT_FCBS
.EXTRN	KILL_BINDEX, KILL_BUFFERS
.EXTRN	NUKE_HEAD_FCB, DEL_EXTFCB
.EXTRN	ARBITRATE_ACCESS
.EXTRN	CONV_ACLOCK, WRITE_DIRTY
.EXTRN	SERIAL_FILE, RELEASE_SERIAL_LOCK
.EXTRN	SWITCH_VOLUME, SEARCH_FCB
.EXTRN	CREATE_FCB, READ_HEADER
.EXTRN	CHECK_PROTECT, WRITE_AUDIT
.EXTRN	GET_MAP_POINTER
.EXTRN	READ_BLOCK, INVALIDATE
.EXTRN	MARK_DIRTY, DELETE_FILE
.EXTRN	CHECKSUM

SE

 03FC 00000  
 08 C2 00002

: 1244

.ENTRY	MARK_DELETE, Save R2,R3,R4,R5,R6,R7,R8,R9
SUBL2	#8, SP

**DELETE  
V04-000**

C 9  
16-Sep-1984 00:15:14 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:30:16 DISK\$VMSMASTER:[F11X.SRC]D

Page 13  
(3)

**DELETE  
V06-010**

D 9  
16-Sep-1984 00:15:14 VAX-11 Bliss-32 v4.0-742 Page 14  
14-Sep-1984 12:30:16 DISK\$VMSMASTER:[F1IX.SRC]DELETE.B32:1 (3)

DE  
VO

0000G	CF	59	DD	0018A	168:	PUSHL	HEADER	#1	CHECKSUM				1499
0000G	CF	01	FB	0018C		CALLS	HEADER						1500
0000G	CF	59	DD	00191		PUSHL	HEADER	#1	MARK_DIRTY				1510
0000G	CF	01	FB	00193	178:	PUSHR	#1	<R2,R3>					1512
		02	BB	00198		CALLS	#2	CONV_ACLOCK					1515
		A3	B5	0019F		TSTW	24	{FCB)					1517
		1B	12	001A2		BNEQ	198						1519
		07	D2	001A4		TSTL	176	(FCB)					1520
		07	13	001A8		BEQL	188						1523
0000G	CF	01	FB	001AC		PUSHL	FCB						1524
0000G	CF	53	DD	001B1	188:	CALLS	#1	KILL_DINDEX					1525
0000G	CF	01	FB	001B3		PUSHL	FCB	#1	DEL_EXTFCB				1526
0000G	CF	53	DD	001B8		PUSHL	FCB						1527
		01	FB	001BA		CALLS	#1	NUKE_HEAD_FCB					1528
		AA	D1	001BF	198:	CMPL	8	(BASE), FCB					1529
		03	12	001C3		BNEQ	208						1530
		08	AA	D4	001C5	CLRL	8	(BASE)					1531
		53	00D0	CA	D1	001C8	208:	CMPL	208	(BASE), FCB			1532
		00D0	7F	12	001CD	BNEQ	258						1533
		00D0	CA	D4	001CF	CLRL	208	(BASE)					1534
		79	11	001D3		BRB	258						1535
30	AA	42	A9	06	28	001D5	218:	MOV3	#6.	66(H HEADER), 48(BASE)			1536
		42	A9	06	28	001D8		MOV3	#6.	66(H HEADER), TEMP_FID			1537
		04	AE	AF	95	001E0		TSTB	TEMP_FID+4				1538
		04	AE	05	12	001E3		BNEQ	228				1539
		04	AE	90	001E5	MOV8	-96	(BASE), TEMP_FID+4					1540
		01	AE	91	001EA	228:	CMP8	TEMP_FID+4, #1					1541
		04	AE	08	12	001EE		BNEQ	238				1542
		04	AA	D5	001F0	TSTL	-96	(BASE)					1543
		04	AE	03	12	001F3		BNEQ	238				1544
		04	AE	94	001F5	CLRB	TEMP_FID+4						1545
		50	69	9A	001F8	238:	MOVZBL	(HEADER), R0					1546
		56	69	9A	001FB	MOVAW	(HEADER)[R0], IDENT_AREA						1547
		66	14	2C	001FF	MOVCS	#20, (IDENT_AREA), #32, #86, (R7)						1548
		66	67		00206								1549
		50	01	A9	9A	00207		MOVZBL	1	(HEADER), R0			1550
		51	69	9A	0020B	MOVZBL	(HEADER), R1						1551
		50	51	C2	0020E	SUBL2	R1, R0						1552
		3C	50	D1	00211	CMPL	R0, #60						1553
		08	08	1F	00214	BLSSU	248						1554
14	A7	36	A6	0042	8F	28	00216	MOV3	#66,	54(IDENT_AREA), 20(R7)			1555
		50	04	AC	D0	0021E		MOVL	F18,	R0			1556
6E	0A	A0	06	29	00222	CMPC3	#6	10(R0), TEMP_FID	258				1557
0056	BF	20	10	BC	0C	25	00227	BNEQ	RESULT_LENGTH,	RESULT, #32, #86, (R7)			1558
		67		AC	2D	00229		CMPC5					1559
		67			00232								1560
		19	12		00233								1561
		42	A9	D4	00235								1562
		46	A9	B4	00238								1563
		40	8F	88	0023B								1564
		59	DD	00240									1565
0000G	CF	01	FB	00242									1566
0000G	CF	59	DD	00247									1567
		01	FB	00249									1568
		50	18	AA	DD	0024E	258:	CALLS	#1	MARK_DIRTY			1569
								MOVL	24	(BASE), R0			1570

DELETE  
V04-000

F 9  
16-Sep-1984 00:15:16 VAX-11 BLISS-32 V4.0-742  
14-Sep-1984 12:30:16 DISK&VMSMASTER:[F1IX.SRC]DELETE.B32;1 Page 16  
(3)

0000G CF	0080 C40 00252	PUSHL 128(BASE)[R0]	:
	01 FB 00257	CALLS #1, WRITE_DIRTY	1562
0000G CF	18 AA DD 0025C	PUSHL 24(BASE)	:
	01 FB 0025F	CALLS #1, RELEASE_SERIAL_LOCK	1564
	18 AA D4 00264	CLRL 24(BASE)	:
	04 00267	RET	1566

; Routine Size: 616 bytes. Routine Base: \$CODES + 006F

```
: 580 1567 1 GLOBAL ROUTINE MARKDEL_FCB (FCB) : L_NORM =
: 581 1568 1
: 582 1569 1
: 583 1570 1
: 584 1571 1
: 585 1572 1
: 586 1573 1
: 587 1574 1
: 588 1575 1
: 589 1576 1
: 590 1577 1
: 591 1578 1
: 592 1579 1
: 593 1580 1
: 594 1581 1
: 595 1582 1
: 596 1583 1
: 597 1584 1
: 598 1585 1
: 599 1586 1
: 600 1587 1
: 601 1588 1
: 602 1589 1
: 603 1590 1
: 604 1591 1
: 605 1592 1
: 606 1593 1
: 607 1594 1
: 608 1595 1
: 609 1596 1
: 610 1597 1
: 611 1598 1
: 612 1599 1
: 613 1600 1
: 614 1601 1
: 615 1602 1
: 616 1603 1
: 617 1604 2 BEGIN
: 618 1605 2
: 619 1606 2 MAP
: 620 1607 2 FCB : REF BBLOCK; ! FCB arg
: 621 1608 2
: 622 1609 2 BIND_COMMON;
: 623 1610 2
: 624 1611 2 EXTERNAL ROUTINE
: 625 1612 2 LOCK_COUNT : L_NORM, ! get count of access locks
: 626 1613 2 QEX_R_CANCEL : L_NORM; ! set fcb8v_stale flag in other fcbs.
: 627 1614 2
: 628 1615 2
: 629 1616 2
: 630 1617 2 If the FCB exists, we mark it for delete (causing the file to be deleted
: 631 1618 2 when the reference count goes to 0). If the
: 632 1619 2 reference count is zero, dump the FCB and its extensions.
: 633 1620 2
: 634 1621 2 IF .FCB NEQ 0
: 635 1622 2 THEN
: 636 1623 3 BEGIN
```

```

637      1626 3   FCB[FCBSV_MARKDEL] = 1;
638      1625 3
639      1626 3
640      1627 3   IF LOCK_COUNT (.FCB [FCBSL_ACCLKID]) NEQ 1
641      1628 3
642      1629 4   THEN
643      1630 4     BEGIN
644      1631 4       IF QEX_N_CANCEL (.FCB [FCBSL_ACCLKID])
645      1632 4       ! Normally the lock will not actually be granted from the qex_n cancel call.
646      1633 4       If it is granted though (success), then set the lockmode field in the
647      1634 4       fcb so that the subsequent conv_acclock handles the value block correctly.
648      1635 4
649      1636 4
650      1637 4   THEN
651      1638 4     FCB [FCBSB_ACCLKMODE] = LCKSK_EXMODE;
652      1639 4
653      1640 4     RETURN 2
654      1641 3
655      1642 3
656      1643 3   IF .FCB[FCBSW_REFCNT] NEQ 0
657      1644 3   THEN
658      1645 3     RETURN 0;           ! file still accessed here
659      1646 3
660      1647 2
661      1648 2
662      1649 2   RETURN 1;           ! ok to delete file
663      1650 2
664      1651 1 END;             ! end of routine MARKDEL_FCB

```

										.EXTRN LOCK_COUNT, QEX_N_CANCEL	
		50	04	0000 00000						.ENTRY MARKDEL_FCB. Save nothing	1567
				AC D0 00002						FCB, R0	1621
22	A0	02	04	39 13 00006						BEQL 38	1625
	50	02	48	02 88 00008						BISB2 #2, 34(R0)	1627
		AC D0 0000C								MOVL FCB, R0	
		40		A0 DD 00010						PUSHL 72(R0)	
0000G	CF	01		01 FB 00015						CALLS #1, LOCK_COUNT	
	01	50		D1 00018						CMPL R0, #1	
		18		18 13 0001B						BEQL 28	
		50	04	AC D0 0001D						MOVL FCB, R0	1630
		48		A0 DD 00021						PUSHL 72(R0)	
0000G	CF	01		01 FB 00024						CALLS #1, QEX_N_CANCEL	
	08	50		E9 00029						BLBC R0, 18	
		50	04	AC D0 0002C						MOVL FCB, R0	1638
OB	A0	05		90 00030						MOVB #5, 11(R0)	
	50	02		D0 00034	18:					MOVL #2, R0	1640
		04		04 00037						RET	
		50	04	AC D0 00038	28:					MOVL FCB, R0	1643
		18		A0 B5 0003C						TSTW 24(R0)	
		04		04 12 0003F						BNEQ 48	
		50	01	D0 00041	38:					MOVL #1, R0	1649
			04	04 00044						RET	
			50	D4 00045	48:					CLRL R0	1651
			04	00047						RET	

DELETE  
V04-000

16-Sep-1984 00:15:16 16-Sep-1984 12:30:16 VAX-11 BLISS-32 V4.0-742  
DISK\$VMSMASTER:[F1IX.SRC]DELETE.B32;1 Page 19 (4)

; Routine Size: 72 bytes, Routine Base: \$CODES + 0207

DE  
VO

: 666 1652 1 ROUTINE DELETE\_HANDLER (SIGNAL, MECHANISM) : L\_NORM =  
: 667 1653 1  
: 668 1654 1 ++  
: 669 1655 1  
: 670 1656 1 FUNCTIONAL DESCRIPTION:  
: 671 1657 1  
: 672 1658 1 This routine is the condition handler for reading the file header.  
: 673 1659 1 If any errors occur, it unwinds and returns to MARK\_DELETE's caller,  
: 674 1660 1 causing the delete of a bad file header to be a quiet NOP.  
: 675 1661 1  
: 676 1662 1  
: 677 1663 1 CALLING SEQUENCE:  
: 678 1664 1 HANDLER (ARG1, ARG2)  
: 679 1665 1  
: 680 1666 1 INPUT PARAMETERS:  
: 681 1667 1 ARG1: address of signal array  
: 682 1668 1 ARG2: address of mechanism array  
: 683 1669 1  
: 684 1670 1 IMPLICIT INPUTS:  
: 685 1671 1 NONE  
: 686 1672 1  
: 687 1673 1 OUTPUT PARAMETERS:  
: 688 1674 1 NONE  
: 689 1675 1  
: 690 1676 1 IMPLICIT OUTPUTS:  
: 691 1677 1 NONE  
: 692 1678 1  
: 693 1679 1 ROUTINE VALUE:  
: 694 1680 1 SSS\_RESIGNAL or none if unwind  
: 695 1681 1  
: 696 1682 1 SIDE EFFECTS:  
: 697 1683 1 NONE  
: 698 1684 1  
: 699 1685 1 --  
: 700 1686 1  
: 701 1687 1  
: 702 1688 2 BEGIN  
: 703 1689 2  
: 704 1690 2 MAP  
: 705 1691 2 SIGNAL : REF BBLOCK; | signal arg array  
: 706 1692 2 MECHANISM : REF BBLOCK; | mechanism arg array  
: 707 1693 2  
: 708 1694 2 BIND\_COMMON;  
: 709 1695 2  
: 710 1696 2 ! If the condition is change mode to user (error exit) cause an unwind to  
: 711 1697 2 return to DELETE's caller.  
: 712 1698 2 Otherwise, just resignal the condition.  
: 713 1699 2 !  
: 714 1700 2  
: 715 1701 2 IF .SIGNAL[CHGOL\_SIG\_NAME] EQL SSS\_CMODUSER  
: 716 1702 2 THEN  
: 717 1703 2 BEGIN  
: 718 1704 2 USER\_STATUS = .SAVE\_STATUS;  
: 719 1705 2 \$UNWIND ();  
: 720 1706 2 END;  
: 721 1707 2  
: 722 1708 2 RETURN SSS\_RESIGNAL; ! status is irrelevant if unwinding

DELETE  
V04-000

: 723  
: 724

1709 2  
1710 1 END;

K 9  
16-Sep-1984 00:15:14  
14-Sep-1984 12:30:16 VAX-11 Bliss-32 v4.0-742  
DISK\$VMSMASTER:[F11X.SRC]DELETE.832;1 Page 21 (5)

: ! end of routine DELETE\_HANDLER

.EXTRN SY\$UNWIND

0000 00000 DELETE_HANDLER:									
									WORD Save nothing
00000424	50	04	AC	D0	00002				MOVL SIGNAL, R0
	8F		A0	D1	00006				CMPL 4(R0), #1060
		04	OE	12	0000E				BNEQ 1\$
	80	AA	C0	AA	D0	00010			MOVL -64(BASE), -128(BASE)
				7E	7C	00015			CLRQ -(SP)
00000006	00		02	FB	00017				CALLS #2, SY\$UNWIND
	50	0918	BF	3C	0001E	1\$:			MOVZWL #2328, R0
				04	00023				RET

: Routine Size: 36 bytes, Routine Base: \$CODE\$ + 031F

: 725 1711 1  
: 726 1712 1 END  
: 727 1713 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	835	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	-----	Symbols	-----	Pages	Processing
	Total	Loaded	Percent	Mapped	Time
\$_255\$DUA28:[SYSLIB]LIB.L32;1	18619	74	0	1000	00:01.9

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:DELETE/OBJ=OBJ\$:DELETE MSRC\$:DELETE/UPDATE=(ENH\$:DELETE)  
: Size: 835 code + 0 data bytes

DELETE  
V04-000

L 9  
16-Sep-1984 00:15:14 VAX-11 Bliss-32 V4.0-742

Page 22

: Run Time: 00:50.6  
: Elapsed Time: 01:48.1  
: Lines/CPU Min: 2033  
: Lexemes/CPU-Min: 56607  
: Memory Used: 352 pages  
: Compilation Complete

0169 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

DEACCS  
LIS

DELETE  
LIS

DIRSNC  
LIS

CREHOR  
LIS

DIRACC  
LIS

CREFCB  
LIS

CREWIN  
LIS

DEL BAD  
LIS

DISPATCH  
LIS      ENTER  
LIS

DELFL  
LIS